Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021 (NetSys 2021)

FlowEmu: An Open-Source Flow-Based Network Emulator

Daniel Stolpmann, Andreas Timm-Giel

4 pages

Guest Editors: Andreas Blenk, Mathias Fischer, Stefan Fischer, Horst Hellbrueck, Oliver Hohlfeld, Andreas Kassler, Koojana Kuladinithi, Winfried Lamersdorf, Olaf Landsiedel, Andreas Timm-Giel, Alexey Vinel
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

FlowEmu: An Open-Source Flow-Based Network Emulator

Daniel Stolpmann[1], **Andreas Timm-Giel**[2]

[1]daniel.stolpmann@tuhh.de, [2]timm-giel@tuhh.de
Hamburg University of Technology, Institute of Communication Networks, Germany

Abstract: When analyzing the impact of a communication system on the Quality of Experience (QoE) of an interactive application, simulation and mathematical modeling typically require reimplementation or modeling of the application under test and only provide insights in terms of selected Key Performance Indicators (KPIs), which raises the need for network emulation. In this demo, we present FlowEmu, an open-source flow-based network emulator that allows the user to manipulate the underlying model and analyze various statistics in real-time via an easy-to-use Graphical User Interface (GUI), by using interactive game streaming as an example application.

Keywords: Network Emulation, Flow-Based Programming, Game Streaming

1 Introduction

Network emulation combines abstract models and real hardware / software components to mimic the behavior of a communication system. Compared to simulation and mathematical modeling, which typically require reimplementation or modeling of the application under test and only provide insights in terms of selected Key Performance Indicators (KPIs), this makes it possible to showcase the impact of a communication system on the Quality of Experience (QoE) of a real interactive application.

In this demo, we present FlowEmu, an open-source flow-based network emulator. FlowEmu provides an extendable set of modules that implement basic functions such as adding delay and introducing packet loss as well as packet queues and departure processes. These modules can be freely connected to model the impact of different types of communication systems on the network traffic of a real application. By providing an easy-to-use web-based Graphical User Interface (GUI), FlowEmu allows the user to change the structure and parameters of the model at runtime. Additionally, it gives insights on internal states by visualizing various statistics.

This paper is structured as follows. In Section 2, existing network emulators are presented and compared to FlowEmu. Section 3 describes the architecture of FlowEmu and highlights some of its features. In Section 4, we give an overview of our demo at NetSys 2021. In the end, we provide a summary in Section 5.

2 Related Work

NetEm [Hem05] is an open-source network emulator that is implemented as a queuing discipline (qdisc) in the Linux kernel and can be set up on any network interface using the “tc” command

line tool. Besides being able to emulate uncorrelated and correlated delay, packet loss, packet corruption, packet duplication and packet re-ordering, it also provides bit rate emulation [Lin]. However, without workarounds, it can only be applied to the outgoing traffic. Also, the included discrete Gilbert-Elliot model is evaluated for each packet, which makes the wall-clock sojourn times dependent on the packet rate. In contrast, FlowEmu provides a continuous-time Gilbert-Elliot model that can be applied to both directions of the traffic simultaneously.

Facebook's Augmented Traffic Control (ATC) [Fac] is based on NetEm and targeted to application developers for testing. It features a web-based GUI and a Representational State Transfer (REST) Application Programming Interface (API) that allows the user to set up and switch between profiles that are parametrized for emulating different types of networks. The project is open-source, but was officially discontinued. Compared to this, FlowEmu is under active development and provides an Message Queuing Telemetry Transport (MQTT) interface as well as a web-based GUI and the option to save and load models.

In [Ste17], a framework for analyzing Active Queue Management (AQM) algorithms was developed. It is build around NetEm and the Linux kernel implementations of different AQM algorithms. The framework supports execution on a physical testbed consisting of multiple computers or a single machine using Docker. In order to get statistics for queuing delay and packet drops, the implementations of the AQM algorithms in the Linux kernel have to be modified to write these statistics into the 16 bit "identification" field of the Internet Protocol Version 4 (IPv4) header. From there, these statistics can be read by the receiving application. FlowEmu also provides the flexibility of running experiments on a testbed or a single machine using Docker, but allows each module to directly record statistics independently of the application under test.

Mahimahi [NSD+15] is an open-source record-and-replay framework for Hypertext Transfer Protocol (HTTP) traffic, which contains a set of network emulation tools called "shells". Each shell creates its own network namespace, in which any program can be executed, and operates on the incoming and outgoing traffic of it. Multiple shells can be chained together by starting them in each other's namespace. Besides shells that are able to add fixed delay or uncorrelated packet loss, there is also a shell that emulates a link based on a trace file, which contains timestamps of packet transmission opportunities. The link shell has an internal queue and is able to create live graphs of the channel capacity, the throughput and the queuing delay. FlowEmu reimplements many of Mahimahi's features and is even able to read its trace files for link emulation. Additionally, it is more flexible due to its modular design and the way modules can be connected while introducing less overhead by running everything in a single process.

3 Architecture and Features

FlowEmu is written in C++ and runs on Linux as a single user space process. The network interfaces are accessed using raw sockets, which provide layer 2 access, so the emulator is able to process the Ethernet Medium Access Control (MAC) header and all higher layer information of the packets.

All functions such as adding delay, introducing packet loss as well as the packet queues and departure processes are implemented in separate modules. Each module has multiple "ports", which are used to connect the modules to each other. Depending on its functionality, a module

can have sending, receiving, requesting or responding ports, which can be connected independently for each direction of a data flow. Packets are passed between modules as references, so no unnecessary copying overhead is introduced. Additionally, each module can define a set of parameters that can be set by the user and a set of statistics to be recorded.

FlowEmu provides an extensive toolchain for running experiments either in a virtual environment or on a physical testbed. Running the emulator as well as the client and server components of the application under test on a single machine in separate Docker containers, which are connected via bridge interfaces, enables fast and convenient development and testing. When running experiments and recording statistics, it is recommended to run the emulator and the application components on separate machines to improve the real-time behavior.

Message Queuing Telemetry Transport (MQTT) Interface FlowEmu is fully controllable via MQTT. This includes changing the structure of the model as well as setting the parameters of the individual modules and getting their statistics. By writing special driver applications, this also allows the emulator to interface with hardware devices, such as Microcontrollers and Musical Instrument Digital Interface (MIDI) controllers.

Graphical User Interface (GUI) FlowEmu provides a web-based GUI that communicates with the emulator via MQTT over a WebSocket. The GUI features a graphical node-based editor, which allows the user to freely connect the modules by drag-and-drop, set the parameters of the individual modules and visualize their statistics.

Runtime Reconfigurable Model FlowEmu applies changes to the structure of the model and the parameters of the individual modules at runtime without resetting the internal state of the modules, which makes it possible to immediately see their effect on the application performance. Besides being an impressive feature for demonstration purposes, this can also be used to quickly test various parameters and find value ranges that are worth to be investigated.

Real-Time Statistics Each module in FlowEmu can provide statistics in real-time, which can give insights on internal states such as queue lengths or can be used if the application under test does not provide end-to-end packet level statistics such as throughput or delay.

Independent and Seedable Pseudo Random Number Generators In FlowEmu, each module uses its own pseudo random number generator, which ensures that the behavior of a module is not influenced by other modules. Additionally, the seed of each pseudo random number generator can be set, which allows results to be reproducible and comparable.

4 NetSys 2021 Demo

At NetSys 2021, we present FlowEmu in a demo. An overview of the setup is shown in Figure 1. The demo is based on a cloud gaming scenario, where a video game is executed on a server and streamed as a video to a client while the inputs of the player are sent back from the client to the server. For this, we use Steam's Remote Play [Val] feature to stream a game from a high-end computer to another device over the network emulator. When changing the structure and parameters of the underlying model via the GUI while the game is being played, the impact of these changes on the performance of the interactive game streaming application can be seen.

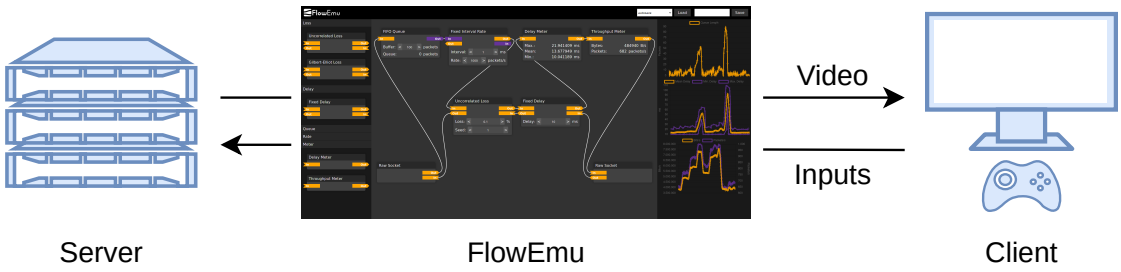

Figure 1: Overview of the demo setup

5 Summary

In this demo, we present FlowEmu, an open-source flow-based network emulator that allows the user to manipulate the underlying model and analyze various statistics in real-time via an easy-to-use GUI and offers advanced features such as independent and seedable pseudo random number generators. For demonstration purposes, we use interactive game streaming as an example application. The source code of FlowEmu is available at [Sto], where we also provide further documentation and invite everyone to contribute improvements and new modules.

Bibliography

[Fac] Facebook. Augmented Traffic Control. (Accessed 12.08.2021). https://github.com/facebookarchive/augmented-traffic-control

[Hem05] S. Hemminger. Network Emulation with NetEm. In *linux.conf.au 2005*. Canberra, Australia, Apr. 2005.

[Lin] Linux Foundation. NetEm. (Accessed 12.08.2021). https://wiki.linuxfoundation.org/networking/netem

[NSD+15] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, H. Balakrishnan. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC '15)*. Pp. 417–429. Santa Clara, CA, USA, July 2015.

[Ste17] H. Steen. Destruction Testing: Ultra-Low Delay using Dual Queue Coupled Active Queue Management. Master's thesis, University of Oslo, Oslo, Norway, 2017.

[Sto] D. Stolpmann. FlowEmu. https://github.com/ComNetsHH/FlowEmu

[Val] Valve Corporation. Steam Remote Play. (Accessed 12.08.2021). https://store.steampowered.com/remoteplay